Discrete-Event Simulation:

A First Course

Section 2.4: Monte Carlo Simulation Examples

Outline

- Overview
- Matrices and Determinants
- Craps
- Hatcheck Girl
- Stochastic Activity Network

Section 2.4: Monte Carlo Simulation Examples

- Recall that *axiomatic* and *experimental* approaches are complementary
- Slight changes in assumptions can sink an axiomatic solution
- In other cases, an axiomatic solution is intractable
- Monte Carlo simulation can be used as an alternative in either case
- Four more examples of MC simulation are presented here

Example 1: Matrices and Determinants

- *Matrix*: set of real or complex numbers in a rectangular array
- For matrix A, a_{ij} is the element in row i, column j

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

Here, A is $m \times n$ — m rows, n columns

- Interesting quantities: eigenvalue, trace, rank, determinant

Determinants

- The *determinant* of a 2×2 matrix A is

$$|A| = \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} = a_{11}a_{22} - a_{21}a_{12}$$

- The determinant of a 3×3 matrix A is

$$\begin{aligned} |A| &= \begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix} \\ &= a_{11} \begin{vmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{vmatrix} - a_{12} \begin{vmatrix} a_{21} & a_{23} \\ a_{31} & a_{33} \end{vmatrix} + a_{13} \begin{vmatrix} a_{21} & a_{22} \\ a_{31} & a_{32} \end{vmatrix} \end{aligned}$$

Random Matrices

- *Random matrix*: matrix whose elements are random variables
- Consider a 3×3 matrix whose elements are random with positive diagonal, negative off-diagonal elements
- Question: What is the probability the determinant is positive?

$$\begin{vmatrix} +u_{11} & -u_{12} & -u_{13} \\ -u_{21} & +u_{22} & -u_{23} \\ -u_{31} & -u_{32} & +u_{33} \end{vmatrix} > 0$$

- Axiomatic solution not easily calculated

  0-13-142917-5

Specification Model

- Let event $\mathcal{A}$ be that the determinant is positive
- Generate N 3×3 matrices with random elements
- Compute the determinant for each matrix
- Let n_a = number of matrices with determinant > 0
- Probability of interest: $\Pr(\mathcal{A}) \cong n_a/N$

  0-13-142917-5

Computational Model: Program det

det

```
for (i = 0; i < N; i++) {
    for (j = 1; j <= 3; j++) {
        for (k = 1; k <= 3; k++) {
            a[j][k] = Random();
            if (j != k)
                a[j][k] = -a[j][k];
        }
    }
    temp1 = a[2][2] * a[3][3] - a[3][2] * a[2][3];
    temp2 = a[2][1] * a[3][3] - a[3][1] * a[2][3];
    temp3 = a[2][1] * a[3][2] - a[3][1] * a[2][2];
    x = a[1][1]*temp1 - a[1][2]*temp2 + a[1][3]*temp3;
    if (x > 0)
        count++;
}
printf(``%11.9f'', (double) count / N);
```

Output From det

- Want N sufficiently large for a good point estimate
- Avoid recycling random number sequences
- Nine calls to `Random()` per 3×3 matrix $\Longrightarrow$ N ¡ m / $9 \cong$ 239 000 000
- For initial seed 987654321 and $N = 200\,000\,000$,

$$\Pr(\mathcal{A}) \cong 0.05017347$$

Point Estimate Considerations

- How many significant digits should be reported?
- Solution: run the simulation multiple times
- One option: Use different initial seeds for each run

  Caveat: Will the same sequences of random numbers appear?
- Another option: Use different a for each run

  Caveat: Use a that gives a good random sequence
- For two runs with $a = 16807$ and 41214

$$\Pr(\mathcal{A}) \cong 0.0502$$

Example 2: Craps

- Toss a pair of fair dice and sum the up faces
- If 7 or 11, win immediately
- If 2, 3, or 12, lose immediately
- Otherwise, sum becomes "point"

  Roll until point is matched (win) or 7 (loss)
- What is $\Pr(\mathcal{A})$, the probability of winning at craps?

Craps: Axiomatic Solution

- Requires conditional probability
- Axiomatic solution: $244/495 \cong 0.493$
- Underlying mathematics must be changed if assumptions change

  E.g., unfair dice

- Axiomatic solution provides a nice consistency check for (easier) Monte Carlo simulation

Craps: Specification Model

- Model one die roll with `Equilikely(1, 6)`

Algorithm 2.4.1

```
wins = 0;
for (i = 1; i <= N; i++) {
    roll = Equilikely(1, 6) + Equilikely(1, 6);
    if (roll == 7 or roll == 11)
        wins++;
    else if (roll != 2 and roll != 3 and roll != 12) {
        point = roll;
        do {
            roll = Equilikely(1, 6) + Equilikely(1, 6);
            if (roll == point) wins++;
        } while (roll != point and roll != 7)
    }
} return (wins/N);
```


Craps: Computational Model

- Program `craps`: uses `switch` statement to determine rolls
- For $N = 10\,000$ and three different initial seeds (see text)

$$\Pr(\mathcal{A}) = 0.497,\ 0.485,\ \text{and}\ 0.502$$

- These results are consistent with 0.493 axiomatic solution
- This (relatively) high probability is attractive to gamblers, yet ensures the house will win in the long run

Example 3: Hatcheck Girl

- Let $\mathcal{A}$ be that all checked hats are returned to wrong owners
- WLOG, let the checked hats be numbered 1, 2, . . . , n
- Girl selects (equally likely) one of the remaining hats to return

$$\Longrightarrow n! \text{ permutations, each with probability } 1/n!$$

- E.g.: When $n = 3$ hats, possible return orders are

1,2,3   1,3,2   2,1,3   2,3,1   3,1,2   3,2,1

- Only 2,3,1 and 3,1,2 correspond to all hats returned incorrectly

$$\Pr(\mathcal{A}) = 1/3$$

Hatcheck: Specification Model

- Generate a random permutation of the first n integers
- The permutation corresponds to the order of hats returned

Clever Shuffling Algorithm (see Section 6.5)

```
for (i = 0; i < n - 1; i++) {
    j = Equilikely(i, n - 1);
    hold = a[j];
    a[j] = a[i]; /* swap a[i] and a[j] */
    a[i] = hold;
}
```

Generates a random permutation of an array a

- Check the permuted array to see if any element matches its index

Hatcheck: Computational Model

- Program `hat`: Monte Carlo simulation of hatcheck problem
- Uses shuffling algorithm to generate random permutation of hats
- For $n = 10$ hats, 10 000 replications, and three different seeds

$$\Pr(\mathcal{A}) = 0.369,\ 0.369,\ \text{and}\ 0.368$$

- What happens to the probability as $n \to \infty$?
- If using simulation, how big should n be?

  Instead, consider axiomatic solution

Hatcheck: Axiomatic Solution

- The probability $\Pr(\mathcal{A})$ of no hat returned correctly is

$$1 - \left(1 - \frac{1}{2!} + \frac{1}{3!} - \cdots + (-1)^{n+1}\frac{1}{n!}\right)$$

- For $n = 10$, $\Pr(\mathcal{A}) \cong 0.36787946$
- Important consistency check for validating `craps`
- As $n \to \infty$, the probability of no hat returned is

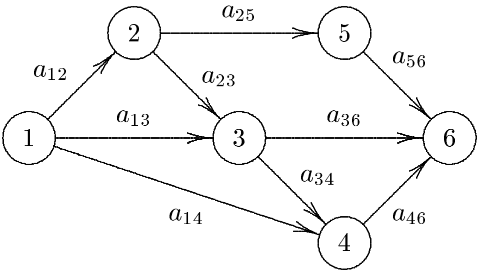
$$1/e \cong 0.36787944$$

Example 4: Stochastic Activity Network

- *Stochastic Activity Network*: network in which arcs represent activities to be completed according to prescribed precedences
- Often used in *project management* — of projects that occur once
- Sequencing of activities is important
- Certain activities cannot begin until others have completed
- *Precedence relationships* establish sequencing between activities

An Example Activity Network

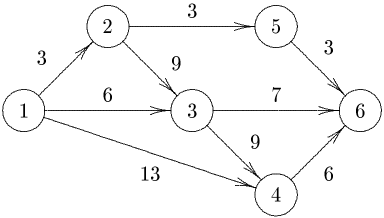
- Arcs represent activities
- Nodes delay the beginning of activities per sequencing constraints



- E.g., activity a_{46} cannot begin until a_{14} and a_{34} have completed

Paths In An Activity Network

- *Path* π_k: ordered sequence of arcs from one node to another
- *Length of* π_k: sum of all activity durations



- Integers along arcs represent time to complete activities
- Question: how long will it take to complete the network?

Critical Paths

- In the previous network, there are $r = 6$ paths

k	Node sequence	π_k	L_k
1	$1 \to 3 \to 6$	$\{a_{13}, a_{36}\}$	13
2	$1 \to 2 \to 3 \to 6$	$\{a_{12}, a_{23}, a_{36}\}$	19
3	$1 \to 2 \to 5 \to 6$	$\{a_{12}, a_{25}, a_{56}\}$	9
4	$1 \to 4 \to 6$	$\{a_{14}, a_{46}\}$	19
5	$1 \to 3 \to 4 \to 6$	$\{a_{13}, a_{34}, a_{46}\}$	21
6	$1 \to 2 \to 3 \to 4 \to 6$	$\{a_{12}, a_{23}, a_{34}, a_{46}\}$	27

- *Critical path* π_c: path with longest length — here, $\pi_c \equiv \pi_6$
- Any path with length < length of π_c can be delayed

Stochastic Activity Networks

- Activity durations are positive random variables
- n nodes, m arcs (activities) in the network
- Single source node (labeled 1), single terminal node (labeled n)
- Y_{ij}: positive random activity duration for arc a_{ij}
- T_j: completion time of all activities entering node j
- A path is critical with a certain probability

$$p(\pi_k) = \Pr(\pi_k \equiv \pi_c), \quad k = 1, 2, \ldots, r$$

SAN: Conceptual Model

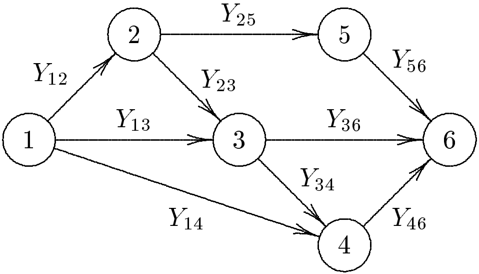
- Represent the network as an $n \times m$ node-arc incidence matrix N

$$N[i,j] = \begin{cases} 1 & \text{arcjleavesnodei} \\ -1 & \text{arcjentersnodei} \\ 0 & \text{otherwise} \end{cases}$$

- Use Monte Carlo simulation to estimate:
    - mean time to complete the network
    - probability that each path is critical

SAN: Conceptual Model

- Each activity duration is a uniform random variate



E.g., Y_{12} has a *Uniform*(0,3) distribution

SAN: Specification Model

- Completion time T_j relates to incoming arcs

$$T_j = \max_{i \in \mathcal{B}(j)} \{T_i + Y_{ij}\} \qquad j = 2, 3, \ldots, n$$

  where $\mathcal{B}(j)$ is the set of nodes immediately before node j

- E.g., in the previous six-node example

$$T_6 = \max\{T_3 + Y_{36}, T_4 + Y_{46}, T_5 + Y_{56}\}$$

- We can write a recursive function to compute the T_j

  0-13-142917-5

SAN: Conceptual Model

- The previous 6-node, 9-arc network is represented as follows:

$$N = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & -1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & -1 \end{bmatrix}$$

- In each row:

  1's represent arcs exiting that node
  -1's represent arcs entering that node

- Exactly one 1 and one -1 in each column

Algorithm 2.4.2

- Returns a random time to complete all activities prior to node j for a single SAN with node-arc incidence matrix N

Algorithm 2.4.2

```
k = 1;
l = 0;
tmax = 0.0;
while (l < |B(j)|) {
    if (N[j][k] == -1) {
        i = 1;
        while (N[j][k] != 1)
            i++;
        t = T_i + Y_i;
        if (t >= tmax) tmax = t;
        l++;
    }
    k++;
}
return (tmax);
```

SAN: Computational Model

- Program `san`: MC simulation of a stochastic activity network
- Uses recursive function to compute completion times T_j (see text)
- Activity durations Y_{ij} are generated at random a priori
- Estimates T_n, the time to complete the entire network
- Computes critical path probabilities $p(\pi_k)$ for $k = 1, 2, \ldots, r$
- Axiomatic approach does not provide an analytic solution

SAN: Computational Model

- For 10 000 realizations of the network and three initial seeds

$$T_6 = 14.64,\ 14.59,\ \text{and}\ 14.57$$

- Point estimates for critical path probabilities are

k	π_k	$\hat{p}_1(\pi_k)$	$\hat{p}_2(\pi_k)$	$\hat{p}_3(\pi_k)$	$\hat{p}_4(\pi_k)$
1	$\{a_{13}, a_{36}\}$	0.0168	0.0181	0.0193	0.0181
2	$\{a_{12}, a_{23}, a_{36}\}$	0.0962	0.0970	0.0904	0.0945
3	$\{a_{12}, a_{25}, a_{56}\}$	0.0013	0.0020	0.0013	0.0015
4	$\{a_{14}, a_{46}\}$	0.1952	0.1974	0.1907	0.1944
5	$\{a_{13}, a_{34}, a_{46}\}$	0.1161	0.1223	0.1182	0.1189
6	$\{a_{12}, a_{23}, a_{34}, a_{46}\}$	0.5744	0.5632	0.5801	0.5726

- Path π_6 is most likely to be critical — 57.26% of the time